Improved Approximation Guarantees for Lower-Bounded Facility Location

(Extended Abstract)

Sara Ahmadian* Chaitanya Swamy*

Abstract

We consider the *lower-bounded facility location* (LBFL) problem (also sometimes called *load-balanced facility location*), which is a generalization of *uncapacitated facility location* (UFL), where each open facility is required to serve a certain *minimum* amount of demand. More formally, an instance $\mathcal{I}$ of LBFL is specified by a set $\mathcal{F}$ of facilities with facility-opening costs $\{f_i\}$, a set $\mathcal{D}$ of clients, and connection costs $\{c_{ij}\}$ specifying the cost of assigning a client j to a facility i, where the c_{ij}s form a metric. A feasible solution specifies a subset F of facilities to open, and assigns each client j to an open facility $i(j) \in F$ so that each open facility serves *at least M clients*, where M is an input parameter. The cost of such a solution is $\sum_{i \in F} f_i + \sum_j c_{i(j)j}$, and the goal is to find a feasible solution of minimum cost.

The current best approximation ratio for LBFL is 448 [18]. We substantially advance the state-of-the-art for LBFL by devising an approximation algorithm for LBFL that achieves a significantly-improved approximation guarantee of 82.6.

Our improvement comes from a variety of ideas in algorithm design and analysis, which also yield new insights into LBFL. Our chief algorithmic novelty is to present an improved method for solving a more-structured LBFL instance obtained from $\mathcal{I}$ via a bicriteria approximation algorithm for LBFL, wherein all clients are aggregated at a subset $\mathcal{F}'$ of facilities, each having at least αM co-located clients (for some $\alpha \in [0, 1]$). One of our key insights is that one can reduce the resulting LBFL instance, denoted $\mathcal{I}_2(\alpha)$, to a problem we introduce, called *capacity-discounted UFL* (CDUFL). CDUFL is a special case of capacitated facility location (CFL) where facilities are either uncapacitated, or have finite capacity and zero opening costs. Circumventing the difficulty that CDUFL inherits the intractability of CFL with respect to LP-based approximation guarantees, we give a simple local-search algorithm for CDUFL based on add, delete, and swap moves that achieves the same approximation ratio (of $1 + \sqrt{2}$) as the corresponding local-search algorithm for UFL. In contrast, the algorithm in [18] proceeds by reducing $\mathcal{I}_2(\alpha)$ to CFL, whose current-best approximation ratio is worse than that of our local-search algorithm for CDUFL, and this is one of the reasons behind our algorithm's improved approximation ratio.

Another new ingredient of our LBFL-algorithm and analysis is a subtly different method for constructing a bicriteria solution for $\mathcal{I}$ (and hence, $\mathcal{I}_2(\alpha)$), combined with the more significant change that we now choose a *random* α from a suitable distribution. This leads to a surprising degree of improvement in the approximation factor, which is reminiscent of the mileage provided by random α-points in scheduling problems.

*{sahmadian,cswamy}@math.uwaterloo.ca. Dept. of Combinatorics and Optimization, Univ. Waterloo, Waterloo, ON N2L 3G1. Supported in part by NSERC grant 327620-09. The second author is also supported by an Ontario Early Researcher Award.

1 Introduction

Facility location problems have been widely studied in the Operations Research community (see, e.g., [13]). In its simplest version, *uncapacitated facility location* (UFL), we are given a set of facilities with opening costs, and a set of clients, and we want to open some facilities and assign each client to an open facility so as to minimize the sum of the facility-opening and client-assignment costs. This problem has a wide range of applications. For example, a company might want to open its warehouses at some locations so that its total cost of opening warehouses and servicing customers is minimized.

We consider the *lower-bounded facility location* (LBFL) problem, which is a generalization of UFL where each open facility is required to serve a certain *minimum* amount of demand. More formally, an LBFL instance $\mathcal{I}$ is specified by a set $\mathcal{F}$ of facilities, and a set $\mathcal{D}$ of clients. Opening facility i incurs a *facility-opening cost* f_i, and assigning a client j to a facility i incurs a *connection cost* c_{ij}. A feasible solution specifies a subset $F \subseteq \mathcal{F}$ of facilities, and assigns each client j to an open facility $i(j) \in F$ so that *each open facility serves at least M clients*, where M is an input parameter. The cost of such a solution is the sum of the facility-opening and connection costs, that is, $\sum_{i \in F} f_i + \sum_j c_{i(j)j}$, and the goal is to find a feasible solution of minimum cost. As is standard in the study of facility location problems, we assume throughout that c_{ij}s form a metric. We use the terms connection cost and assignment cost interchangeably in the sequel.

LBFL can be motivated from various perspectives. This problem was introduced independently by Karger and Minkoff [8], and Guha, Meyerson, and Munagala (who called the problem *load-balanced facility location*) [5] (see also [3]), both of whom arrived at LBFL as a means of solving their respective buy-at-bulk style network design problems. LBFL arises as a natural subroutine in such settings because obtaining a near-optimal solution to the buy-at-bulk problem often entails aggregating a certain minimum demand at certain hub locations, and then connecting the hubs via links of lower per-unit-demand cost (and higher fixed cost). LBFL also finds direct applications in supply-chain logistics problems, where the lower-bound constraint can be used to model the fact that it is not profitable or feasible to use services unless they satisfy a certain minimum demand. For example (as noted in [18]), Lim, Wang, and Xu [11], use LBFL to abstract a transportation problem faced by a company that has to determine the allocation of cargo from customers to carriers, who then ship their cargo overseas. Here the lower bound arises because each carrier, if used, is required (by regulation) to deliver a minimum amount of cargo. Also, LBFL is an interesting special case of *universal facility location* (UniFL) [12]—a generalization of UFL where the facility cost depends on the number of clients served by it—with non-increasing facility-cost functions. UniFL with arbitrary non-increasing functions is not a well-understood problem, and the study of LBFL may provide useful insights here.

Clearly, LBFL with $M = 1$ is simply UFL, and hence, is *NP*-hard; consequently, we are interested in designing approximation algorithms for LBFL. The first constant-factor approximation algorithm for LBFL was devised by Svitkina [18], whose approximation ratio is 448. Prior to this, the only known approximation guarantees were *bicriteria guarantees*. [8] and [5] independently devised (ρ, α)-approximation algorithms via a reduction to UFL: these algorithms return a solution of cost at most ρ times the optimum where each open facility serves at least αM clients ($\alpha < 1$, ρ is a function of α).

Our results and techniques. We devise an approximation algorithm for LBFL that achieves a substantially-improved approximation guarantee of 82.6 (Theorem 3.1), thus significantly advancing the state-of-the-art for LBFL. Our improvement comes from a combination of ideas in algorithm design and analysis, and yields new insights about the approximability of LBFL. In order to describe the ideas underlying our improvement, we first briefly sketch Svitkina's algorithm.

Svitkina's algorithm begins by using the reduction in [8, 5] to obtain a bicriteria solution for $\mathcal{I}$, which is then used to convert $\mathcal{I}$ into an LBFL instance $\mathcal{I}_2$ with facility-set $\mathcal{F}' \subseteq \mathcal{F}$ having the following structure: (i) all clients are aggregated at $\mathcal{F}'$ with each facility $i \in \mathcal{F}'$ having $n_i \geq \alpha M$ co-located clients; (ii) all facilities in $\mathcal{F}'$ have zero opening costs; and (iii) near-optimal solutions to $\mathcal{I}_2$ translate to near-optimal solutions to

$\mathcal{I}$ (and vice versa). The goal now is to identify a subset of $\mathcal{F}'$ to close, such that transferring the clients aggregated at these closed facilities to the remaining (open) facilities in $\mathcal{F}'$ ensures that each remaining facility serves at least M demand (and the cost incurred is "small"). [18] shows that one can achieve this by solving a suitable CFL instance. Essentially the idea is that a facility i that remains open corresponds to a *demand point* in the CFL instance that requires $M - n_i$ units of demand, and a facility i that is closed maps to a *supply point* in the CFL instance having n_i units that can be supplied to demand points (i.e., open facilities). Of course, one does not know beforehand which facilities will be closed and which will remain open; so to encode this correspondence in the CFL instance, we create at every location $i \in \mathcal{F}'$, a supply point with (suitable opening cost and) capacity M, and a demand point with demand $M - n_i$ if $n_i \leq M$ (so the supply point at i has n_i residual capacity after satisfying this demand). (Assume $n_i \leq M$ for simplicity; facilities with $n_i > M$ are treated differently.) Finally, [18] argues that a CFL-solution (where a supply point may end up sending *less* then n_i supply to other demand points) can be mapped to a solution to $\mathcal{I}_2$ without increasing the cost incurred by much; since CFL admits an $O(1)$-approximation algorithm, one obtains an $O(1)$-approximate solution to $\mathcal{I}_2$, and hence to the original LBFL instance $\mathcal{I}$.

Our algorithm also proceeds by (a) obtaining an LBFL instance $\mathcal{I}_2$ satisfying properties (i)–(iii) mentioned above, (b) solving $\mathcal{I}_2$, and (c) mapping the $\mathcal{I}_2$-solution to a solution to $\mathcal{I}$, but our implementation of steps (a) and (b) differs from that in Svitkina's algorithm. These modified implementations, which are independent of each other and yield significant improvements in the overall approximation ratio even when considered in isolation, result in our much-improved approximation ratio. We detail how we perform step (a) later, and focus first on describing how we solve $\mathcal{I}_2$, which is our chief algorithmic contribution.

Our key insight is that one can solve the LBFL instance $\mathcal{I}_2$ by reducing it to a new problem we introduce that we call *capacity-discounted* UFL (CDUFL), which closely resembles UFL and admits an algorithm (that we devise) with a much better approximation ratio than CFL. A CDUFL-instance has the property that every facility is either uncapacitated (i.e., has infinite capacity), or has finite capacity and *zero* facility cost. The CDUFL instance we construct consists of the same supply and demand points as in the reduction of $\mathcal{I}_2$ to CFL in [18], except that all supply points with non-zero opening cost are now uncapacitated. (An interesting consequence is that if all facilities in $\mathcal{I}_2$ have $n_i \leq M$, the CDUFL instance is in fact a UFL-instance!)

We prove two crucial algorithmic results. It is not hard to see that the "standard" integrality-gap example for the natural LP-relaxation of CFL can be cast as a CDUFL instance, thus showing that the natural LP-relaxation for CDUFL has a large integrality gap (see Appendix A); in fact, we are not aware of any LP-relaxation for CDUFL with constant integrality gap. Circumventing this difficulty, we devise a local-search algorithm for CDUFL based on add, swap, and delete moves that achieves the *same performance guarantees* as the corresponding local-search algorithm for UFL [1] (see Section 4.2). The local-search algorithm yields significant dividends in the overall approximation ratio because not only is its approximation ratio for CDUFL better than the state-of-the-art for CFL, but also because it yields separate (asymmetric) guarantees on the facility-opening and assignment costs, which allows one to perform a tighter analysis. Second, we show that any near-optimal CDUFL-solution can be mapped to a near-optimal solution to $\mathcal{I}_2$ (see Section 4.1). As before, it could be that in the CDUFL-solution, a supply point i (which corresponds to facility i being closed down) sends less than n_i supply to other demand points, so that closing down i entails transferring its residual clients to open facilities. But since some supply points are now uncapacitated, it could also be that i sends more than n_i supply to other demand points. We argue that this artifact can also be handled without increasing the solution cost by much, by opening the facilities in a carefully-chosen subset of $\{i\} \cup \{\text{demand points satisfied by } i\}$ and closing down the remaining facilities. For *every value of* α (recall that the LBFL instance $\mathcal{I}_2$ is specified in terms of a parameter α), the resulting approximation factor for $\mathcal{I}_2$ (Theorem 3.5) is better than the guarantee obtained for $\mathcal{I}_2$ in Svitkina's algorithm; this in turn translates (by choosing α suitably) to an improved solution to the original instance.

We now discuss how we implement step (a), that is, how we obtain instance $\mathcal{I}_2$. As in [18], we arrive at $\mathcal{I}_2$ by computing a bicriteria solution to LBFL, but we obtain this bicriteria solution in a different fashion

(see Section 3). The reduction from LBFL to UFL in [8, 5] proceeds by setting the opening cost of facility i to $f_i + \frac{2\alpha}{1-\alpha} \cdot \sum_{j \in \mathcal{D}(i)} c_{ij}$, where $\mathcal{D}(i)$ is the set of M clients closest to i, solving the resulting UFL instance, and postprocessing using (single-facility) delete moves if such a move improves the solution cost. We modify this reduction subtly by creating a UFL instance, where facility i's opening cost is instead set to $f_i + 2\alpha M R_i(\alpha)$, where $R_i(\alpha)$ is the distance between i and the αM-closest client to it. As in the case of the earlier reduction, we argue that each open facility i in the resulting solution (obtained by solving UFL and postprocessing) serves at least αM clients. The overall bound we obtain on the total cost now includes various $R_i(\alpha)$ terms. Instead of plugging in the (weak) bound $MR_i(\alpha) \leq \frac{\sum_{j \in \mathcal{D}(i)} c_{ij}}{1-\alpha}$ (which would yield the same guarantee as that obtained via the earlier reduction), we are able to perform a tighter analysis by choosing α from a suitable distribution and leveraging the fact that $M \int_0^1 R_i(\alpha) d\alpha = \sum_{j \in \mathcal{D}(i)} c_{ij}$. (This can easily be derandomized, since there are only M combinatorially distinct choices for α.) These simple modifications (in algorithm-design *and* analysis) yield a surprising amount of improvement in the approximation factor, which is reminiscent of the mileage provided by (random) α-points for various scheduling problems (see, e.g., [16]) and UFL [15, 17]. Also, we observe that one can obtain further improvements by using the local-search algorithm of [2, 1] to solve the above UFL instance: this is because the resulting solution is then already postprocessed, which allows us to exploit the asymmetric bounds on the facility-opening and assignment costs provided by the local-search algorithm via scaling, and improve the approximation ratio.

Finally, we remark that the study of CDUFL may provide useful and interesting insights about CFL. CDUFL is a special case of CFL that despite its special structure inherits the intractability of CFL with respect to LP-based approximation guarantees. If one seeks to develop LP-based techniques and algorithms for CFL (which has been a long-standing and intriguing open question), then one needs to understand how one can leverage LP-based techniques for CDUFL, and it is plausible that LP-based insights developed for CDUFL may yield similar insights for CFL (and potentially LP-based approximation guarantees for CFL).

Related work. As mentioned earlier, LBFL was independently introduced by [8] and [5], who used it as a subroutine to solve the (*rent-or-buy* and hence, the) *maybecast* problem, and the *access network design* problem respectively. Their ideas, which lead to bicriteria guarantees for LBFL, play a preprocessing role both in Svitkina's algorithm for LBFL [18] and (slightly indirectly) in our algorithm.

There is a large body of literature that deals with approximation algorithms for (metric) UFL, CFL and its variants; see [14] for a survey on UFL. The first constant approximation guarantee for UFL was obtained by Shmoys, Tardos, and Aardal [15] via an LP-rounding algorithm, and the current state-of-the-art is a 1.488-approximation algorithm due to Li [10]. Local-search techniques have also been utilized to obtain $O(1)$-approximation guarantees for UFL [9, 2, 1]. We apply some of the ideas of [2, 1] in our algorithm. Starting with the work of Korupolu, Plaxton, and Rajaraman [9], various local-search algorithms with constant approximation ratios have been devised for CFL, with the current-best approximation ratio being $5.83 + \epsilon$ [19]. Local-search approaches are however not known to work for LBFL; in Appendix B, we show that local search based on add, delete, and swap moves yields poor approximation guarantees. Universal facility location (UniFL), where the facility cost is a non-decreasing function of the number of clients served by it, was introduced by [6, 12], and [12] gave a constant approximation algorithm for this. We are not aware of any work on UniFL with arbitrary non-increasing functions (which generalizes LBFL). [4] give a constant approximation for the case where the cost-functions do not decrease too steeply (the constant depends on the steepness); notice that LBFL does not fall into this class so their results do not apply here.

2 Problem definition and notation

Recall that we have a set $\mathcal{F}$ of facilities with facility-opening costs $\{f_i\}$, a set $\mathcal{D}$ of clients, metric connection (or assignment) costs $\{c_{ij}\}$ specifying the cost of assigning client j to facility i, and a (integer) parameter

M. Our objective is to open a subset F of facilities and assign each client j to an open facility $i(j) \in F$, so that at least M clients are assigned to each open facility, and the total cost incurred, $\sum_{i\in F} f_i + \sum_j c_{i(j)j}$, is minimized. We use $\mathcal{I}$ to denote this LBFL instance.

Let F^* and C^* denote respectively the facility-opening and assignment cost of an optimal solution to $\mathcal{I}$; we will often refer to this solution as "the optimal solution" in the sequel. We sometimes abuse notation and also use F^* to denote the set of open facilities in this optimal solution. Let $OPT = F^* + C^*$ denote the total optimal cost. For a facility $i \in \mathcal{F}$, let $\mathcal{D}(i)$ be the set of M clients closest to i, and $R_i(\alpha)$ denote the distance between i and the $\lceil \alpha M \rceil$-closest client to i; that is, if $\mathcal{D}(i) = \{j_1, \ldots, j_M\}$, where $c_{ij_1} \leq \ldots \leq c_{ij_M}$, then $R_i(\alpha) = c_{ij_{\lceil \alpha M \rceil}}$ (for $0 < \alpha \leq 1$). Let $R^*(\alpha) = \sum_{i\in F^*} R_i(\alpha)$. Observe that each $R_i(\alpha)$ is an increasing function of α, $M\int_0^1 R_i(\alpha)d\alpha = \sum_{j\in\mathcal{D}(i)} c_{ij}$, and $R_i(\alpha) \leq \left(\sum_{j\in\mathcal{D}(i)} c_{ij}\right)/(M - \lceil \alpha M \rceil + 1) \leq \frac{\sum_{j\in\mathcal{D}(i)} c_{ij}}{M(1-\alpha)}$. Hence, $R^*(\alpha)$ is an increasing function of α, $M\int_0^1 R^*(\alpha)d\alpha \leq C^*$, and $R^*(\alpha) \leq \frac{C^*}{M(1-\alpha)}$.

3 Our algorithm and the main theorem

We now give a high-level description of our algorithm using certain building blocks that are supplied in the subsequent sections. Let $\mathcal{I}$ denote the LBFL instance.

(1) **Obtaining a bicriteria solution.** Construct a UFL instance with the same set of facilities and clients, and the same assignment costs as $\mathcal{I}$, where the opening cost of facility i is set to $f_i + 2\alpha M R_i(\alpha)$. Use the local-search algorithm for UFL in [2] or [1] with scaling parameter $\gamma > 0$ to solve this UFL instance. (We set α, γ suitably to get the desired approximation; see Theorem 3.1.) Let $\mathcal{F}' \subseteq \mathcal{F}$ be the set of facilities opened in the UFL-solution. Claim 3.2 and Lemma 3.3 show that each $i \in F'$ serves at least αM clients.

(2) **Transforming to a structured LBFL instance.** We use the bicriteria solution obtained above to transform $\mathcal{I}$ into another structured LBFL instance $\mathcal{I}_2$ as in [18]. In the instance $\mathcal{I}_2$, we set the opening cost of each $i \in \mathcal{F}'$ to zero, and we "move" to i all the $n_i \geq \alpha M$ clients assigned to it, that is, all these clients are now co-located at i. So $\mathcal{I}_2$ consists of only the points in $\mathcal{F}'$ (which forms both the facility-set and client-set). We will sometimes use the notation $\mathcal{I}_2(\alpha)$ to indicate explicitly that $\mathcal{I}_2$'s specification depends on the parameter α.

(3) Solve $\mathcal{I}_2$ using the method described in Section 4. Obtain a solution to $\mathcal{I}$ by opening the same facilities and making the same client assignments as in the solution to $\mathcal{I}_2$.

Analysis. Our main theorem is as follows.

Theorem 3.1 *For any $\alpha \in (0.5, 1]$ and $\gamma > 0$, the above algorithm returns a solution to $\mathcal{I}$ of cost at most*

$$F^*\big(1 + \gamma h(\alpha)\big) + C^*\Big(2h(\alpha) - 1 + \tfrac{2}{\gamma}\Big) + 2\gamma\alpha M R^*(\alpha)h(\alpha) + 2\alpha M R^*(\alpha)$$

where $h(\alpha) = 1 + \frac{4}{\alpha} + \frac{4\alpha}{2\alpha-1} + 4\sqrt{\frac{6}{2\alpha-1}}$. Thus, we can compute efficiently a solution to $\mathcal{I}$ of cost at most:

(i) $92.84 \cdot OPT$, by setting $\alpha = 0.75, \gamma = 3/h(\alpha)$;

(ii) $82.6 \cdot OPT$, by letting γ be a suitable (efficiently-computable) function of α, and choosing α randomly from the interval $[0.67, 1]$ according to the density function $p(x) = \frac{1}{\ln(1/0.67)x}$.

The roadmap for proving Theorem 3.1 is as follows. We first bound the cost of the bicriteria solution obtained in step (1) in terms of OPT (Lemma 3.3). This will allow us to bound the cost of an optimal solution to $\mathcal{I}_2$, and argue that mapping an $\mathcal{I}_2$-solution to a solution to $\mathcal{I}$ does not increase the cost by much (Lemma 3.4). The only missing ingredient is a guarantee on the cost of the solution to $\mathcal{I}_2$ found in step (3), which we supply in Theorem 3.5, whose proof appears in Section 4.

The following claim follows from essentially the same arguments as in [8, 5].

Claim 3.2 *Let S' be a* delete-optimal *solution to the above UFL instance; that is, the total UFL-cost does not decrease by deleting any open facility of S'. Then, each facility of S' serves at least αM clients.*

The local-search algorithms for UFL in [2, 1] have the same performance guarantees and both include a delete-move as a local-search operation, so upon termination, we obtain a delete-optimal solution.[1] Observe that opening the same facilities and making the same client assignments as in the optimal solution to $\mathcal{I}$ yields a solution S to the UFL instance constructed in step (1) of the algorithm with facility cost $F^S \leq F^* + 2\alpha MR^*(\alpha)$ and assignment cost $C^S \leq C^*$. Combined with the analysis in [2, 1], this yields the following. (For simplicity, we assume that all local-search algorithms return a local optimum; standard arguments show that dropping this assumption increases the approximation by at most a $(1+\epsilon)$ factor.)

Lemma 3.3 *For a given parameter $\gamma > 0$, executing the local-search algorithm in [2, 1] on the above UFL instance returns a solution with facility cost F_b and assignment cost C_b satisfying $F_b \leq F^* + 2\alpha MR^*(\alpha) + 2C^*/\gamma,\ C_b \leq \gamma\big(F^* + 2\alpha MR^*(\alpha)\big) + C^*$, where each open facility serves at least αM clients.*

Lemma 3.4 ([18]) *(i) The (assignment) cost $C^*_{\mathcal{I}_2}$ of an optimal solution to $\mathcal{I}_2$ is at most $2(C_b + C^*)$.*
(ii) Any solution to $\mathcal{I}_2$ of cost C yields a solution to $\mathcal{I}$ of cost at most $F_b + C_b + C$.

Theorem 3.5 *For any $\alpha > 0.5$, there is a $g(\alpha)$-approximation algorithm for $\mathcal{I}_2(\alpha)$, where $g(\alpha) = \frac{2}{\alpha} + \frac{2\alpha}{2\alpha-1} + 2\sqrt{\frac{2}{\alpha^2} + \frac{4}{2\alpha-1}}$.*

Remark 3.6 Our $g(\alpha)$-approximation ratio for $\mathcal{I}_2(\alpha)$ improves upon the approximation obtained in [18] by a factor of roughly 2 *for all* α. Thus, plugging in our algorithm for solving $\mathcal{I}_2$ in the LBFL-algorithm in [18] (and choosing a suitable α), already yields an improved approximation factor of 218 for LBFL.

Proof of Theorem 3.1 : Recall that $h(\alpha) = 1 + \frac{4}{\alpha} + \frac{4\alpha}{2\alpha-1} + 4\sqrt{\frac{6}{2\alpha-1}}$. Note that $2g(\alpha) + 1 \leq h(\alpha)$ for all $\alpha \in [0, 1]$; we use this upper bound throughout below. Combining Theorem 3.5 and the bounds in Lemmas 3.3 and 3.4, we obtain a solution to $\mathcal{I}$ of cost at most $F_b + \big(2g(\alpha)+1\big)C_b + 2g(\alpha)C^*$

$$
\begin{aligned}
&\leq \quad F^* + 2\alpha MR^*(\alpha) + \frac{2C^*}{\gamma} + h(\alpha)\gamma\Big(F^* + 2\alpha MR^*(\alpha)\Big) + \big(2h(\alpha)-1\big)C^* \\
&= \quad F^*\big(1+\gamma h(\alpha)\big) + C^*\Big(2h(\alpha) - 1 + \tfrac{2}{\gamma}\Big) + 2\gamma\alpha MR^*(\alpha)h(\alpha) + 2\alpha MR^*(\alpha).
\end{aligned}
$$

Part (i) follows by plugging in the values of α and γ, and using the bound $R^*(\alpha) \leq \frac{C^*}{M(1-\alpha)}$.

Let $\beta = 0.67$. For part (ii), we set $\gamma = \frac{K}{\sqrt{h(\alpha)}}$, where $K = \left(\ln^2(1/\beta) \cdot \mathrm{E}_\alpha\left[h(\alpha)\right] / \big(\frac{\int_\beta^1 h(x)dx}{1-\beta}\big)\right)^{\frac{1}{4}}$. Plugging in this γ, we see that the cost incurred is at most

$$
F^*\big(1 + K\sqrt{h(\alpha)}\big) + C^*\Big(2h(\alpha) - 1 + \tfrac{2}{K}\sqrt{h(\alpha)}\Big) + 2K\alpha MR^*(\alpha)\sqrt{h(\alpha)} + 2\alpha MR^*(\alpha).
$$

We now bound the expected cost incurred when one chooses α randomly according to the stated density function. This will also yield an explicit expression for K (as a function of β), thus showing that K (and hence, γ) can be computed efficiently. We note that $\mathrm{E}\left[\sqrt{X}\right] \leq \sqrt{\mathrm{E}\left[X\right]}$ and utilize Chebyshev's Integral inequality (see [7]): if f and g are non-increasing and non-decreasing functions respectively from $[a, b]$ to

[1] A subtle point is that typically local-search algorithms terminate only with an "approximate" local optimum. However, one can then execute all delete moves that improve the solution cost, and thereby obtain a delete-optimal solution.

$\mathbb{R}_+$, then $\int_a^b f(x)g(x)dx \leq \frac{(\int_a^b f(x)dx)(\int_a^b g(x)dx)}{b-a}$. Observe that $h(\alpha)$ decreases with α. Recall that $\beta = 0.67$. We have the following.

$$\begin{aligned} \mathrm{E}_\alpha\left[h(\alpha)\right] &= c_2(\beta) := \Big[\frac{4}{\beta} - 4 + 8\sqrt{6}\big(\pi/4 - \tan^{-1}(\sqrt{2\beta-1})\big) + 2\ln\Big(\frac{1}{2\beta-1}\Big) + \ln(1/\beta)\Big]/\ln(1/\beta) \\ \mathrm{E}_\alpha\left[\alpha M R^*(\alpha)\right] &= M\Big(\int_\beta^1 R^*(x)dx\Big)/\ln(1/\beta) \leq C^*/\ln(1/\beta). \end{aligned}$$

Finally, using Chebyshev's inequality, we obtain that

$$\mathrm{E}_\alpha\left[\alpha M R^*(\alpha)\sqrt{h(\alpha)}\right] \leq \Big[M\Big(\int_\beta^1 R^*(x)dx\Big)\frac{\int_\beta^1 dx\sqrt{h(x)}}{1-\beta}\Big]/\ln(1/\beta) \leq \Big[C^*\sqrt{c_3(\beta)}\Big]/\ln(1/\beta),$$

where

$$c_3(\beta) := \big(\int_\beta^1 h(x)dx\big)/(1-\beta) = \Big[4\ln(1/\beta) + 4\sqrt{6}\big(1-\sqrt{2\beta-1}\big) + 3(1-\beta) + \ln\Big(\frac{1}{2\beta-1}\Big)\Big]/(1-\beta).$$

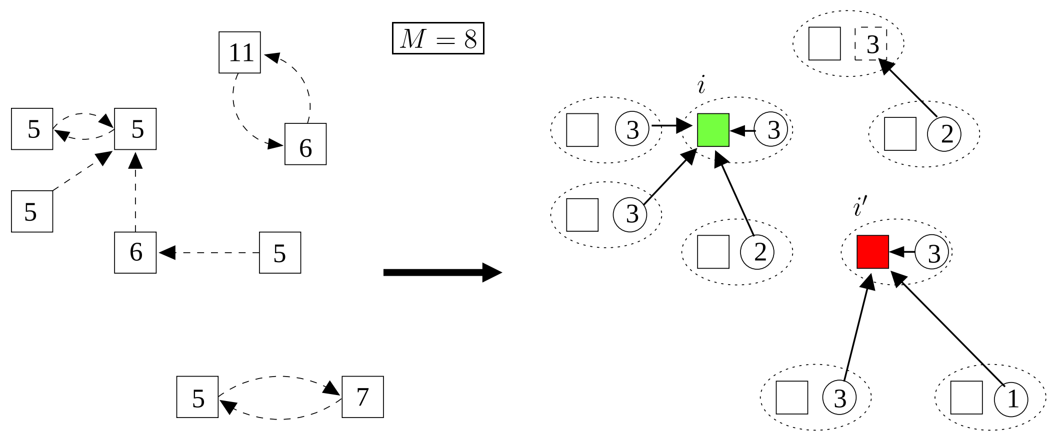
The second inequality follows since $\big(\int_\beta^1 dx\sqrt{h(x)}\big)/(1-\beta) = \mathrm{E}_{\alpha\sim\text{uniform in }[\beta,1]}\Big[\sqrt{h(\alpha)}\Big]$. Plugging in these bounds, we get that $K = \big(\ln^2(1/\beta)c_2(\beta)/c_3(\beta)\big)^{0.25}$ and the total cost is at most

$$F^*\Big(1 + \big(\tfrac{\ln^2(1/\beta)(c_2(\beta))^3}{c_3(\beta)}\big)^{\frac{1}{4}}\Big) + C^*\Big(2c_2(\beta) - 1 + 4\big(\tfrac{c_2(\beta)c_3(\beta)}{\ln^2(1/\beta)}\big)^{\frac{1}{4}} + \tfrac{2}{\ln(1/\beta)}\Big) < 82.59(F^* + C^*). \qquad \blacksquare$$

4 Solving instance $\mathcal{I}_2(\alpha)$

We now describe our algorithm for solving instance $\mathcal{I}_2(\alpha)$ and analyze its performance guarantee, thereby proving Theorem 3.5. As mentioned earlier, one of the key differences between our algorithm and the one in [18] is that instead of reducing $\mathcal{I}_2$ to capacitated facility location (CFL), we solve $\mathcal{I}_2$ by reducing it to a new problem that we call *capacity-discounted UFL* (CDUFL). CDUFL is a special case of CFL where all facilities with non-zero opening cost are uncapacitated (i.e., have infinite capacity). Perhaps surprisingly, despite this special structure, CDUFL inherits the intractability of CFL with respect to LP-based approximation guarantees: there is no known LP-relaxation for CDUFL that has constant integrality gap; Appendix A shows that the natural LP-relaxation for CDUFL has bad integrality gap. However, as we show in Section 4.2, we can obtain a simple local-search algorithm for CDUFL whose approximation ratio is better than the current-best approximation for CFL.

Recall that $\mathcal{I}_2$ has only the points in $\mathcal{F}' \subseteq \mathcal{F}$, and there are $n_i \geq \alpha M$ co-located clients at each $i \in \mathcal{F}'$. Let $l(i) = \min_{i'\in\mathcal{F}', i'\neq i} c_{ii'}$. To avoid confusion, we refer to the facilities and clients in the CDUFL instance as supply points and demand points respectively. The CDUFL instance created to solve $\mathcal{I}_2$ resembles the CFL instance created in [18]; the difference is that all supply points with non-zero opening costs are now uncapacitated. More precisely, at each $i \in \mathcal{F}'$, we create an uncapacitated supply point with opening cost $\delta\min\{n_i, M\}l(i)$, where δ is a parameter we fix later. If $n_i > M$ we create a second supply point at i with capacity $n_i - M$ and zero opening cost. If $n_i < M$, we create a demand point at i with demand $M - n_i$. Let $\mathcal{I}'$ denote this CDUFL instance (see Fig. 1). Let $\mathcal{F}^u$, $\mathcal{F}^c$ denote respectively the set of uncapacitated and capacitated supply points of $\mathcal{I}'$. Roughly speaking, satisfying a demand point i by non-co-located supply points translates to leaving facility i open in the $\mathcal{I}_2$ solution; hence, its demand is set to $M - n_i$, which is the number of additional clients it needs. Conversely, opening the uncapacitated supply point at i and supplying demand points from i translates to closing i in the $\mathcal{I}_2$ solution and transferring its co-located clients to other open facilities.



(a) $\mathcal{I}_2$

(b) $\mathcal{I}'$, and a solution S for $\mathcal{I}'$

Figure 1: (a) An $\mathcal{I}_2$ instance. Each box denotes a facility, and the number inside the box is the number of co-located clients; a dashed arrow $i \to i'$ denotes that i' is the closest facility to i.
(b) The corresponding $\mathcal{I}'$ instance. The boxes and circles represent supply points and demand points respectively, and points inside a dotted oval are co-located. A solid box denotes an uncapacitated supply point, and a dashed box denotes a capacitated facility whose capacity is shown inside the box. The number inside a circle is the demand of that demand point. The arrows indicate a solution S to $\mathcal{I}'$, where i and i' are the two open uncapacitated supply points.

Lemma 4.1 ([18]) *There exists a solution to $\mathcal{I}'$ with facility cost $F \le \delta C^*_{\mathcal{I}_2}$ and assignment cost $C \le C^*_{\mathcal{I}_2}$.*

Theorem 4.2 *(i) Given any CDUFL instance, one can efficiently compute a solution with facility-opening cost $\widehat{F} \le F^{\text{sol}} + 2C^{\text{sol}}$ and assignment cost $\widehat{C} \le F^{\text{sol}} + C^{\text{sol}}$, where F^{sol} and C^{sol} are the facility and assignment costs of an arbitrary solution to the CDUFL instance.*
(ii) Thus, Lemma 4.1 implies that one can compute a solution to $\mathcal{I}'$ with facility cost $F_{\mathcal{I}'}$ and assignment cost $C_{\mathcal{I}'}$ satisfying $F_{\mathcal{I}'} \le (2+\delta)C^*_{\mathcal{I}_2},\ C_{\mathcal{I}'} \le (1+\delta)C^*_{\mathcal{I}_2}$.

We defer the description of the local-search algorithm for CDUFL, and the proof of Theorem 4.2 to Section 4.2. We first describe how to convert an $\mathcal{I}'$-solution to a solution to $\mathcal{I}_2$ with a small increase in cost, and show how this combined with Theorem 4.2 leads to the approximation bound for $\mathcal{I}_2$ stated in Theorem 3.5.

4.1 Mapping an $\mathcal{I}'$-solution to an $\mathcal{I}_2$-solution

An $\mathcal{I}'$-solution need not directly translate to an $\mathcal{I}_2$ solution because an open supply point i may not supply (and hence, transfer) exactly n_i units of demand (see, e.g., i and i' in Fig. 1(b)). Since we have uncapacitated supply points, we have to consider both the cases where i supplies more than n_i demand (a situation not encountered in [18]), and less than n_i demand. Suppose that we are given a solution S to $\mathcal{I}'$ with facility cost F^S and assignment cost C^S (see Fig. 1(b)). Again, we abuse notation and use F^S to also denote the set of supply points that are opened in S. Let N_i initialized to n_i keep track of the number of clients at location $i \in \mathcal{F}'$. Our goal is to reassign clients (using S as a template) so that at the end we have $N_i = 0$ or $N_i \ge M$ for each $i \in \mathcal{F}'$. Observe that once we have determined which facilities in $\mathcal{F}'$ will have $N_i \ge M$ (i.e., the facilities to open in the $\mathcal{I}_2$-solution), one can find the best way of (re)assigning clients by solving a min-cost flow problem. However, for purposes of analysis, it will often be convenient to explicitly specify a (possibly suboptimal) reassignment. We may assume that: (i) $\mathcal{F}^c \subseteq F^S$; (ii) if S opens an uncapacitated supply point located at some $i \in \mathcal{F}'$ with $n_i > M$, then the demand assigned to the capacitated supply

point at i equals its capacity $n_i - M$; (iii) for each $i \in \mathcal{F}'$ with $n_i \leq M$, if the supply point at i is open then it serves the entire demand of the co-located demand point; and (iv) at most one *uncapacitated* supply point serves, maybe partially, the demand of any demand point; we say that this uncapacitated supply point satisfies the demand point. We reassign clients in three phases.

A1. **(Removing capacitated supply points)** Consider any location $i \in \mathcal{F}'$ with $n_i > M$. Let i^1 and i^2 denote respectively the capacitated and uncapacitated supply points located at i. If i^1 supplies x units to the demand point at location i', we transfer x clients from location i to i'. Now if i^1 has $y > 0$ leftover units of capacity in S, then we "move" y clients to i^2 (which is not open in S). We update the N_is accordingly. Note that this reassignment effectively gets rid of all capacitated supply points. Thus, there is now exactly one uncapacitated supply point and at most one demand point at each location $i \in \mathcal{F}'$; we refer to these simply as supply point i and demand point i below.

Let X_i be the total demand from other locations assigned to supply point i. Let $\mathcal{F}^G = \{i \in \mathcal{F}' : N_i < X_i\}$, $\mathcal{F}^R = \{i \in \mathcal{F}' : N_i \geq X_i > 0\}$, and $\mathcal{F}^B = \{i \in \mathcal{F}' : X_i = 0\}$. which is the set of supply points that are not opened in S. Note that $N_i \geq \min\{n_i, M\} \geq \alpha M$ for all $i \in \mathcal{F}'$, and $N_i = \min\{n_i, M\}$ for all $i \in \mathcal{F}^R \cup \mathcal{F}^G$ (because of properties (ii) and (iii) above).

A2. **(Taking care of $\mathcal{F}^R$ and demand points satisfied by $\mathcal{F}^R$)** For each $i \in \mathcal{F}^R$, if i supplies x units to demand point i', we move x clients from i to i', and update $N_i, N_{i'}$. We now have $N_i = \min\{n_i, M\} - X_i$ residual clients at each $i \in \mathcal{F}^R$, which we must reduce to 0, or increase to at least M. We follow the same procedure as in [18], which we sketch below.

For each $i \in \mathcal{F}^R$, we include an edge (i, i') where $i' \in \mathcal{F}'$ is the facility nearest to i (recall that $c_{ii'} = l(i)$). We use an arbitrary but fixed tie-breaking rule here, so each component of the resulting digraph is a directed tree rooted at either (i) a node $r \in \mathcal{F}' \setminus \mathcal{F}^R$, or (ii) a 2-cycle $(r, r'), (r', r)$, where $r, r' \in \mathcal{F}^R$. We break up each component Γ into a collection of smaller components as follows. Essentially, we move the residual clients of supply points in the component bottom-up from the leaves up to the root, cut off the component at the first node u that accumulates at least M clients, and recurse on the portion of the component not containing u. More precisely, let Γ_u denote the subtree of Γ rooted at node $u \in \Gamma$ (if u belongs to a 2-cycle then we do not include the other node of this 2-cycle in Γ_u).

- If $\sum_{i \in \Gamma} N_i < M$, or if Γ is of type (i) and all children u of the root satisfy $\sum_{i \in \Gamma_u} N_i < M$, we leave Γ unchanged.
- Otherwise, let u be a deepest (i.e., furthest from root) node in Γ such that $\sum_{i \in \Gamma_u} N_i \geq M$. We delete the arc leaving u. If this disconnects u from $\Gamma \setminus \Gamma_u$, then we recurse on $\Gamma \setminus \Gamma_u$.
- Otherwise u must belong to the root 2-cycle of Γ. Let r' be the other node of this 2-cycle. If $\sum_{i \in \Gamma_{r'}} N_i \geq M$, we delete r''s outgoing arc (thus splitting Γ into Γ_u and $\Gamma_{r'}$).

After applying the above procedure (to all components), if we are left with a component of type (ii) with $\sum_{i \in \text{ component}} N_i \geq M$, we convert it to type (i) by arbitrarily deleting one of the arcs of the 2-cycle. Thus, at the end of this process, we have two types of components.

(a) A tree T rooted at a node r: we move the N_i residual clients of each non-root node $i \in T$ to r.

(b) A type-(ii) tree T with root $\{r, r'\}$: we must have $\sum_{i \in T} N_i < M$. Let $i' \in \mathcal{F}^B$ be the location nearest to $\{r, r'\}$; we move the N_i residual clients of each $i \in T$ to i'.

Update the N_is to reflect the above reassignment. Observe that we now have $N_i = 0$ or $N_i \geq M$ for each $i \in \mathcal{F}^R$, and each $i \in \mathcal{F}^B$ has $n_i \geq M$, or is a demand point satisfied by a supply point in $\mathcal{F}^G$. Figure 2(a) shows a snapshot after steps A1 and A2 have been executed on the solution shown in Fig. 1(b). Here $i' \in \mathcal{F}^R$ has one client left after moving clients to the bottom two facilities, which is then transferred to i_3.

A3. **(Taking care of $\mathcal{F}^G$ and demand points satisfied by $\mathcal{F}^G$)** For $i \in \mathcal{F}^G$, let $D(i)$ be the set of demand points $j \in \mathcal{F}'$, $j \neq i$ satisfied by i, and let $D'(i) = \{j \in D(i) : N_j < M\}$. Note that $D(i) \subseteq \mathcal{F}^B$. Phase A2 may only increase N_j for all j in $\mathcal{F}^B \cup \mathcal{F}^G$, so $N_j \geq \alpha M$ for all $j \in \mathcal{F}^G \cup \big(\bigcup_{i \in \mathcal{F}^G} D(i)\big)$.

Fix $i \in \mathcal{F}^G$. We reassign clients so that $N_j = 0$ or $N_j \geq M$ for all $j \in \{i\} \cup D'(i)$, without decreasing N_j for $j \in D(i) \setminus D'(i)$. Applying this procedure to all supply points in $\mathcal{F}^G$ will complete our task. Define $Y_j = M - N_j$ (which is at most $M - n_j$) for $j \in D'(i)$. We consider two cases.

- $\sum_{j \in D'(i)} Y_j \leq N_i$. For each $j \in D'(i)$, if i supplies x units to j, we transfer x clients from i to j. If i is now left with less than M residual clients, we move these residual clients to the location in $D(i)$ nearest to i.
- $\sum_{j \in D'(i)} Y_j > N_i$ (see Fig. 2). Let $i_0 = i$, and $D'(i) = \{i_1, \ldots, i_t\}$, where $c_{i_1 i} \leq \ldots \leq c_{i_t i}$. Let $\ell = t - \Big\lfloor \frac{\sum_{r=0}^t N_{i_r}}{M} \Big\rfloor = \Big\lceil \frac{\sum_{r=1}^t Y_{i_r} - N_{i_0}}{M} \Big\rceil$, so $\ell \geq 1$ (and $\ell < t$ since $N_{i_0} + N_{i_1} \geq M$). Note that ℓ is the unique index such that $\sum_{r=\ell+1}^t Y_{i_r} \leq \sum_{r=0}^{\ell} N_{i_r} < \sum_{r=\ell+1}^t Y_{i_r} + M$. This enables us to transfer Y_{i_q} clients to each i_q, $q = \ell+1, \ldots, t$ from the locations $i_\ell, \ldots, i_0$—we do this by transferring all clients of i_r (where $1 \leq r \leq \ell$) before considering i_{r-1}—and be left with at most M residual clients in $\{i_0, \ldots, i_\ell\}$. We argue that these residual clients are all concentrated at i_0 and i_1, with i_1 having at most $(1-\alpha)M$ residual clients. We transfer these residual clients to $i_{\ell+1}$.

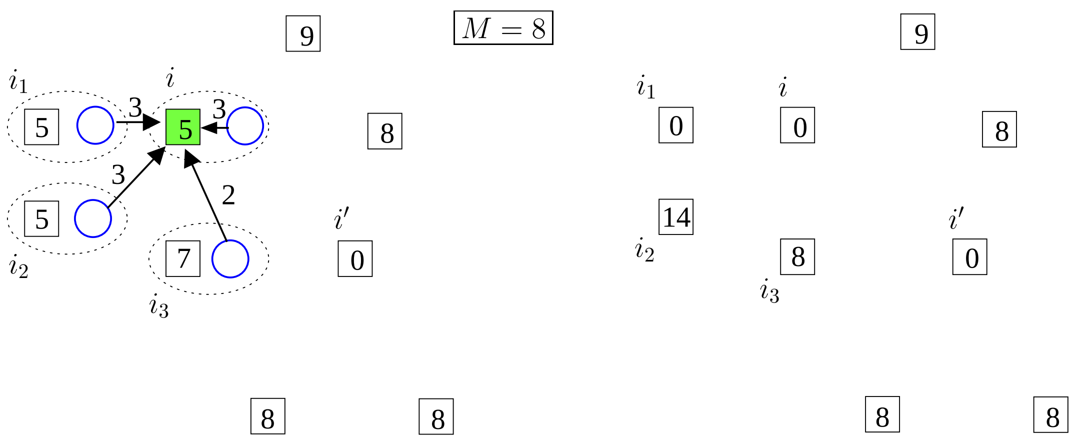


Figure 2: The number inside a box is the current value of N_i; the number labeling an arrow is the demand assignment of the $\mathcal{I}'$-solution. The circles indicate demand points j with $N_j < M$. (a) The situation after running steps A1 and A2 on the solution in Fig. 1(b). (b) The situation after running step A3.

Theorem 4.3 *The above algorithm returns an $\mathcal{I}_2$-solution of cost at most $\frac{F^S}{\delta\alpha} + C^S\big(\frac{1}{\alpha} + \frac{2\alpha}{2\alpha-1}\big)$. Thus, taking S to be the solution mentioned in part (ii) of Theorem 4.2, and $\delta = \sqrt{\frac{2/\alpha}{1/\alpha + (2\alpha)/(2\alpha-1)}}$, we obtain a solution to $\mathcal{I}_2(\alpha)$ satisfying the approximation bound stated in Theorem 3.5.*

Proof : Let S_2 denote the solution computed for $\mathcal{I}_2$. For a supply point i opened in S, we use C_i^S to denote the cost incurred in supplying demand from i to the demand points satisfied by i; so $C^S = \sum_{i \in F^S} C_i^S$. At various steps, we transfer clients between locations according to the assignment in the CDUFL solution S, and the cost incurred in this reassignment can be charged against the C_i^Ss of the appropriate supply points. So the cost of phase A1 is $\sum_{i \in \mathcal{F}^c} C_i^S$, and the cost of the first step of phase A2 is $\sum_{i \in \mathcal{F}^R} C_i^S$.

As in [18], we can bound the remaining cost of phase A2, incurred in transferring clients according to the tree edges by $F^S/\delta\alpha+\bigl(\sum_{i\in\mathcal{F}^R}C_i^S\bigr)/(2\alpha-1)$. When we move clients up to the root of a component, we move strictly less than M clients along any edge (i,i') in that component, and since $i\in\mathcal{F}^R$, we pay at least $\delta\alpha Ml(i)$ opening cost for i. The only unaccounted cost now is the cost incurred in step (b) of phase A2, where we have a tree T rooted at $\{r,r'\}$. Let $i'\in\mathcal{F}^B$ be the location nearest to $\{r,r'\}$, and (say) $c_{i'r}\leq c_{i'r'}$. Note that we have already bounded the cost in transferring clients to r, so we only need to bound the cost incurred in transferring at most M clients from r to i'. This is at most $M\cdot\frac{C_r^S+C_{r'}^S}{X_r+X_{r'}}\leq\bigl(C_r^S+C_{r'}^S\bigr)/(2\alpha-1)$, because $\{r,r'\}$ send $X_r+X_{r'}=(n_r+n_{r'})-(N_r+N_{r'})\geq(2\alpha-1)M$ units to demand points in $\mathcal{F}^B$, all of which are at distance at least $c_{i'r}$ from $\{r,r'\}$.

Finally, consider phase A3 and some $i\in\mathcal{F}^G$. If $\sum_{j\in D'(i)}Y_j\leq N_i$, then the cost incurred is at most $C_i^S+M\cdot\frac{C_i^S}{X_i}\leq C_i^S\bigl(1+\frac{1}{\alpha}\bigr)$ (as $X_i>N_i\geq\alpha M$). Now consider the case $\sum_{j\in D'(i)}Y_j>N_i$. For any $i_q\in\{i_{\ell+1},\dots,i_t\}$ and any $i_r\in\{i_0,\dots,i_\ell\}$, we have $c_{i_ri_q}\leq 2c_{ii_q}$, so the cost of transferring $Y_{i_q}\leq M-n_{i_q}$ clients to each $i_q,\ q=\ell+1,\dots,t$ is at most $2C_i^S$. Observe that $(t-\ell+1)M>\sum_{r=0}^t N_{i_r}$, i.e., $M+\sum_{q=\ell+1}^t Y_{i_r}>\sum_{r=0}^{\ell}N_{i_r}$, so after this reassignment, there are less than M residual clients in $i_0,\dots,i_\ell$. By our order of transferring clients, all these residual clients are at i_0,i_1 (otherwise we would have at least $N_{i_0}+N_{i_1}\geq M$ residual clients) with at most $M-N_{i_0}\leq(1-\alpha)M$ of them located at i_1. The cost of reassigning these residual clients is at most $(1-\alpha)Mc_{ii_1}+Mc_{ii_{\ell+1}}\leq(1-\alpha)M\cdot\frac{C_i^S}{\sum_{r=1}^t Y_{i_r}}+M\cdot\frac{C_i^S}{\sum_{r=\ell+1}^t Y_{i_r}}$, since C_i^S is the total cost of supplying at least Y_{i_r} demand to each $i_r,\ r=1,\dots,t$. The latter expression is at most $C_i^S\bigl(\frac{1-\alpha}{\alpha}+\frac{1}{2\alpha-1}\bigr)$, since $\sum_{r=1}^t Y_{i_r}>N_{i_0}\geq\alpha M$, $\sum_{r=\ell+1}^t Y_{i_r}>\sum_{r=0}^{\ell}N_{i_r}-M\geq(2\alpha-1)M$.) Thus, the cost of S_2 is at most

$$\frac{F^S}{\delta\alpha}+\sum_{i\in\mathcal{F}^c}C_i^S+\sum_{i\in\mathcal{F}^R}C_i^S\cdot\Bigl(1+\tfrac{1}{2\alpha-1}\Bigr)+\sum_{i\in\mathcal{F}^G}C_i^S\cdot\max\Bigl\{1+\tfrac{1}{\alpha},2+\tfrac{1-\alpha}{\alpha}+\tfrac{1}{2\alpha-1}\Bigr\}\leq\frac{F^S}{\delta\alpha}+C^S\Bigl(\tfrac{1}{\alpha}+\tfrac{2\alpha}{2\alpha-1}\Bigr).$$

So if S is the solution given by part (ii) of Theorem 4.2, the cost of S_2 is at most $\bigl(\frac{2}{\delta\alpha}+\frac{1}{\alpha}+(1+\delta)(\frac{1}{\alpha}+\frac{2\alpha}{2\alpha-1})\bigr)C^*_{\mathcal{I}_2}$, and plugging in the value of δ yields the $g(\alpha)=\frac{2}{\alpha}+\frac{2\alpha}{2\alpha-1}+2\sqrt{\frac{2}{\alpha^2}+\frac{4}{2\alpha-1}}$ approximation bound stated in Theorem 3.5. ■

4.2 A local-search based approximation algorithm for CDUFL

We now describe our local-search algorithm for CDUFL, which leads to the proof of Theorem 4.2. Let $\widehat{\mathcal{F}}=\widehat{\mathcal{F}}^u\cup\widehat{\mathcal{F}}^c$ be the facility-set of the CDUFL instance, where $\widehat{\mathcal{F}}^u\cap\widehat{\mathcal{F}}^c=\emptyset$. Here, $\widehat{\mathcal{F}}^u$ are the uncapacitated facilities with opening costs $\{\widehat{f}_i\}$, and facilities in $\widehat{\mathcal{F}}^c$ have (finite) capacities $\{u_i\}$ and zero opening costs. Let $\widehat{\mathcal{D}}$ be the set of clients and $\widehat{c}_{ij}$ be the cost of assigning client j to facility i. The goal is to open facilities and assign clients to open facilities (respecting the capacities) so as to minimize the sum of the facility-opening and client-assignment costs. We can find the best assignment of clients to open facilities by solving a network flow problem, so we focus on determining the set of facilities to open.

The local-search algorithm consists of three moves: $\mathrm{add}(i')$, $\mathrm{delete}(i)$, $\mathrm{swap}(i,i')$, which respectively, add a facility i' not currently open, delete a facility i that is currently open, and swap facility i that is open with facility i' that is not open. We note that *all* previous (local-search) algorithms for CFL that work with non-uniform capacities use moves that are more complicated than the moves above (and involve adding and/or deleting multiple facilities at a time). The algorithm repeatedly executes the best cost-improving move (if one exists) until no such move exists. (As mentioned earlier, to ensure polynomial time, we only consider moves that yield significant improvement and hence terminate at an approximate local optimum; but this has only a marginal effect on the approximation bound.) We assume for simplicity that each client has unit demand. This is without loss of generality because, even with non-unit client-demands, one can

compute the best local-search move (and hence run the algorithm), and for the purposes of analysis, one can always treat a client with integer demand d as d co-located unit-demand clients.

Analysis. Let $\widehat{S}$ denote a local-optimum returned by the algorithm, with facility-opening cost (and set of open facilities) $\widehat{F}$ and assignment cost $\widehat{C}$. Let sol be an arbitrary CDUFL solution, with facility-cost (and set of open facilities) F^{sol} and assignment cost C^{sol}. Note that we may assume that $\widehat{\mathcal{F}}^c \subseteq \widehat{F} \cap F^{\mathrm{sol}}$. For a facility i, we use $\widehat{\mathcal{D}}_{\widehat{S}}(i)$ and $\widehat{\mathcal{D}}_{\mathrm{sol}}(i)$ to denote respectively the (possibly empty) set of clients served by i in $\widehat{S}$ and sol. For a client j, let $\widehat{C}_j$ and C_j^{sol} be the assignment cost of j in $\widehat{S}$ and sol respectively.

We borrow ideas from the analysis of the corresponding local-search algorithm for UFL in [1], but the presence of capacitated facilities means that we need to reassign clients more carefully to analyze the change in assignment cost due to a local-search move. In particular, unlike the analysis in [1], where upon deletion of a facility $s \in \widehat{F}$ we reassign only the clients currently assigned to s, in our case (as in the analysis of local-search algorithms for CFL), we need to perform a more "global" reassignment (i.e., even clients not assigned to s may get reassigned) along certain (possibly long) paths in a suitable graph. This also means that we need to construct a suitable mapping between paths instead of the client-mapping considered in [1].

We construct a directed graph G with node-set $\widehat{\mathcal{D}} \cup \widehat{\mathcal{F}}$, and arcs from i to all clients in $\widehat{\mathcal{D}}_{\widehat{S}}(i)$ and arcs from all clients in $\widehat{\mathcal{D}}_{\mathrm{sol}}(i)$ to i, for every facility i. Via standard flow-decomposition, we can decompose G into a collection of (simple) paths $\mathcal{P}$, and cycles $\mathcal{R}$, so that (i) each facility i appears as the starting point of $\max\{0, |\widehat{\mathcal{D}}_{\widehat{S}}(i)| - |\widehat{\mathcal{D}}_{\mathrm{sol}}(i)|\}$ paths, and the ending point of $\max\{0, |\widehat{\mathcal{D}}_{\mathrm{sol}}(i)| - |\widehat{\mathcal{D}}_{\widehat{S}}(i)|\}$ paths, and (ii) each client j appears on a unique path P_j or on a cycle. Let $\mathcal{P}^{\mathsf{st}}(s) \subseteq \mathcal{P}$ and $\mathcal{P}^{\mathsf{end}}(o) \subseteq \mathcal{P}$ denote respectively the collection of paths starting at s and ending at o, and $\mathcal{P}(s,o) = \mathcal{P}^{\mathsf{st}}(s) \cap \mathcal{P}^{\mathsf{end}}(o)$. For a path $P = \{i_0, j_0, i_1, j_1, \ldots, i_k, j_k, i_{k+1} := o\} \in \mathcal{P}$, define $\widehat{\mathcal{D}}(P) = \{j_0, \ldots, j_k\}$, $head(P) = j_0$, and $tail(P) = j_k$. A *shift* along P means that we reassign client j_r to i_{r+1} for each $r = 0, \ldots, k$ (opening o if necessary). Note that this is feasible, since if $o \in \widehat{\mathcal{F}}^c$, we know that $|\widehat{\mathcal{D}}_{\widehat{S}}(o)| \leq |\widehat{\mathcal{D}}_{\mathrm{sol}}(o)| - 1 \leq u_o - 1$. Let $shift(P) := \sum_{j \in \widehat{\mathcal{D}}(P)} \bigl(C_j^{\mathrm{sol}} - \widehat{C}_j\bigr)$ be the increase in assignment cost due to this reassignment, which is an upper bound on the actual increase in assignment cost if o is added to $\widehat{F}$. Also, let $cost(P) := \sum_{j \in \widehat{\mathcal{D}}(P)} \bigl(C_j^{\mathrm{sol}} + \widehat{C}_j\bigr)$. We define a shift along a cycle $R \in \mathcal{R}$ similarly, letting $shift(R) := \sum_{j \in \widehat{\mathcal{D}} \cap R} \bigl(C_j^{\mathrm{sol}} - \widehat{C}_j\bigr)$. By considering a shift operation for every path and cycle in $\mathcal{P} \cup \mathcal{R}$ (i.e., suitable add moves), we get the following result.

Lemma 4.4 *For every $o \in F^{\mathrm{sol}}$ and any $\mathcal{Q} \subseteq \mathcal{P}^{\mathsf{end}}(o)$, we have $\sum_{P \in \mathcal{Q}} shift(P) \geq \begin{cases} -\widehat{f}_o & \text{if } o \in F^{\mathrm{sol}} \setminus \widehat{F}, \\ 0 & \text{otherwise}. \end{cases}$ For every cycle $R \in \mathcal{R}$, we have $shift(R) \geq 0$. Thus, we have $\widehat{C} \leq F^{\mathrm{sol}} + C^{\mathrm{sol}}$.*

Bounding the opening cost of facilities in $\widehat{F} \setminus F^{\mathrm{sol}}$. For this, we only need paths that start at facility in $\widehat{F} \setminus F^{\mathrm{sol}}$. Note that all facilities in $(\widehat{F} \setminus F^{\mathrm{sol}}) \cup (F^{\mathrm{sol}} \setminus \widehat{F})$ are *uncapacitated*. To avoid excessive notation, for a facility $o \in F^{\mathrm{sol}} \setminus \widehat{F}$, we now use $\mathcal{P}^{\mathsf{end}}(o)$ to refer to the collection of paths ending in o that start in $\widehat{F} \setminus F^{\mathrm{sol}}$. (As before, $\mathcal{P}(s,o)$ is the set of paths that start at s and end at o.) For any $o \in F^{\mathrm{sol}} \setminus \widehat{F}$, we can obtain a 1-1 mapping $\pi : \mathcal{P}^{\mathsf{end}}(o) \mapsto \mathcal{P}^{\mathsf{end}}(o)$ such that if $P \in \mathcal{P}(s,o),\ s \in \widehat{F} \setminus F^{\mathrm{sol}}$ and $\pi(P) = P' \in \mathcal{P}(s',o)$, then (i) if $|\mathcal{P}(s,o)| \leq \frac{|\mathcal{P}^{\mathsf{end}}(o)|}{2}$, we have $s \neq s'$; (ii) if $s = s'$, then $P = P'$; and (iii) $\pi(P') = P$. Say that $o \in F^{\mathrm{sol}} \setminus \widehat{F}$ is *captured* by s if $|\mathcal{P}(s,o)| > \frac{|\mathcal{P}^{\mathsf{end}}(o)|}{2}$. Note that o is captured by at most one facility in $\widehat{F}$. Call a facility in $\widehat{F} \setminus F^{\mathrm{sol}}$ *good* if it does not capture any facility, and *bad* otherwise.

Lemma 4.5 *For any good facility s, we have*

$$\widehat{f}_s \leq \sum_{P \in \mathcal{P}^{\mathsf{st}}(s)} shift(P) + \sum_{o \notin \widehat{F}, P \in \mathcal{P}(s,o)} cost\bigl(\pi(P)\bigr). \tag{1}$$

Proof : Consider the move $\mathrm{delete}(s)$. We upper bound the increase in reassignment cost as follows. Consider $j \in \widehat{\mathcal{D}}_{\widehat{S}}(s)$, and let $P_j \in \mathcal{P}(s,o)$. (Recall that P_j is the unique path containing j.) If $o \in \widehat{F} \cap F^{\mathrm{sol}}$, then we perform a shift along P_j. Otherwise, let $\pi(P_j) \in \mathcal{P}(s',o)$, where $s' \neq s$. We reassign all clients on P_j except $tail(P_j)$ as in the shift operation, and reassign $tail(P_j)$ to s'. Let $k = tail(P_j)$. Since $c_{s'k} \leq c_{s'o} + C_k^{\mathrm{sol}} \leq cost\big(\pi(P_j)\big) + C_k^{\mathrm{sol}}$, the increase in cost by reassigning clients on P_j this way is at most $cost\big(\pi(P_j)\big) + C_k^{\mathrm{sol}} - \widehat{C}_k + \sum_{j' \in \widehat{\mathcal{D}}(P_j)\setminus\{k\}}\big(C_{j'}^{\mathrm{sol}} - \widehat{C}_{j'}\big)$. Thus, the actual increase in cost due to this move, which should be nonnegative, is at most

$$-\widehat{f}_s + \sum_{o \in \widehat{F}, P \in \mathcal{P}(s,o)} shift(P) + \sum_{o \notin \widehat{F}, P \in \mathcal{P}(s,o)} \Big[shift(P) + cost\big(\pi(P)\big)\Big].$$

■

Now consider a bad facility s. Let $\mathrm{capt}_s \subseteq F^{\mathrm{sol}} \setminus \widehat{F}$ be the facilities captured by s, and let $o_s \in \mathrm{capt}_s$ be the facility nearest to s.

Lemma 4.6 *For any bad facility s, we have*

$$\widehat{f}_s \leq \sum_{o \in \mathrm{capt}_s} \widehat{f}_o + \sum_{P \in \mathcal{P}^{\mathsf{st}}(s)} shift(P) + \sum_{\substack{o \notin \widehat{F} \\ P \in \mathcal{P}(s,o): \pi(P) \neq P}} cost\big(\pi(P)\big) + \sum_{\substack{o \in \mathrm{capt}_s \setminus \{o_s\} \\ P \in \mathcal{P}(s,o): \pi(P) = P}} cost(P). \tag{2}$$

Proof : Consider the move $\mathrm{swap}(s, o_s)$. We reassign client $j \in \widehat{\mathcal{D}}_{\widehat{S}}(s)$ as follows. Let $P_j \in \mathcal{P}(s,o)$.

- If $o \in \widehat{F} \cap F^{\mathrm{sol}}$, or $o = o_s$ and $\pi(P_j) = P_j$, we perform a shift along P_j. The increase in assignment cost is at most $shift(P_j)$.

  Otherwise, let $\pi(P_j) \in \mathcal{P}(s', o)$.
- If $\pi(P_j) \neq P_j$ (so $s' \neq s$), we reassign $\widehat{\mathcal{D}}(P_j) \setminus \{tail(P_j)\}$ as in the shift operation, and assign $tail(P_j)$ to s'. As in the proof of Lemma 4.5, the increase in assignment cost is at most $shift(P_j) + cost\big(\pi(P_j)\big)$.
- If $\pi(P_j) = P_j$ (so $o \neq o_s$), we assign j to o_s. Note that $c_{o_s j} \leq \widehat{C}_j + c_{s o_s} \leq \widehat{C}_j + c_{so} \leq \widehat{C}_j + cost(P_j)$, so the increase in assignment cost is at most $cost(P_j)$.

This gives the inequality

$$\begin{aligned} 0 \leq \widehat{f}_{o_s} - \widehat{f}_s &+ \sum_{\substack{P \in \mathcal{P}(s,o): o \in \widehat{F} \text{ or} \\ o = o_s,\ \pi(P) = P}} shift(P) + \sum_{o \notin \widehat{F}} \sum_{P \in \mathcal{P}(s,o): \pi(P) \neq P} \Big[shift(P) + cost\big(\pi(P)\big)\Big] \\ &+ \sum_{o \notin \widehat{F}: o \neq o_s} \sum_{P \in \mathcal{P}(s,o): \pi(P) = P} cost(P). \end{aligned} \tag{3}$$

Now consider the operation $\mathrm{add}(o)$ for all $o \in \mathrm{capt}_s \setminus \{o_s\}$, and apply Lemma 4.4 taking $\mathcal{Q} = \{P \in \mathcal{P}(s,o) : \pi(P) = P\}$. This yields the inequality $0 \leq \widehat{f}_o + \sum_{P \in \mathcal{P}(s,o): \pi(P) = P} shift(P)$ for each $o \in \mathrm{capt}(s) \setminus \{o_s\}$. Adding these inequalities to (3), and rearranging proves the lemma. ■

Proof of Theorem 4.2 : We prove part (i); part (ii) follows directly from part (i) and Lemma 4.1. Lemma 4.4 bounds $\widehat{C}$. Consider adding (1) for all good facilities and (2) for all bad facilities, and the vacuous equality $\widehat{f}_i = \widehat{f}_i$ for all $i \in \widehat{F} \cap F^{\mathrm{sol}}$. The LHS of the resulting inequality is precisely $\widehat{F}$. The $\widehat{f}_i$s on the RHS add up to give at most F^{sol}. We claim that each path $P \in \bigcup_{s \in \widehat{F} \setminus F^{\mathrm{sol}}} \mathcal{P}^{\mathsf{st}}(s)$ contributes at most $shift(P) + cost(P) = 2\sum_{j \in \widehat{\mathcal{D}}(P)} C_j^{\mathrm{sol}}$ to the RHS. Thus the RHS is at most $F^{\mathrm{sol}} + 2C^{\mathrm{sol}}$, and we obtain that $\widehat{F} \leq F^{\mathrm{sol}} + 2C^{\mathrm{sol}}$.

Each path P in $\bigcup_{s\notin F^{\mathrm{sol}},o\in\widehat{F}}\mathcal{P}(s,o)$ appears exactly once, either in (1) or in (2), and contributes $\mathit{shift}(P)$. Now consider a path $P\in\bigcup_{s\notin F^{\mathrm{sol}},o\notin\widehat{F}}\mathcal{P}(s,o)$, and let $\pi(P)=P'\in\mathcal{P}(s',o)$. Note that $\pi(P')=P$. If $P'\neq P$, then P appears twice in our inequality-system: once in the inequality for s contributing $\mathit{shift}(P)$ (due to P), and once in the inequality for s' contributing $\mathit{cost}(P)$ (due to P'). If $P'=P$, then $s=s'$ and s is a bad facility; now P appears only in (2) (for s) and contributes either $\mathit{shift}(P)$ if $o=o_s$, or $\mathit{shift}(P)+\mathit{cost}(P)$ otherwise. ■

Corollary of Theorem 4.2: *There is a $\left(1+\sqrt{2}\right)$-approximation algorithm for CDUFL.*

Proof : We take sol in part (i) of Theorem 4.2 to be an optimum solution (with cost $F^{\mathrm{opt}}+C^{\mathrm{opt}}$) to the instance, and scale the facility costs by σ before running the local-search algorithm. The solution returned has cost $F+C\leq\left(F^{\mathrm{opt}}+\frac{2}{\sigma}\cdot C^{\mathrm{opt}}\right)+\left(\sigma F^{\mathrm{opt}}+C^{\mathrm{opt}}\right)$. Setting $\sigma=\sqrt{2}$ yields the result. ■

References

[1] V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004.

[2] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[3] S. Guha, A. Meyerson, and K. Munagala. A constant factor approximation for the single sink edge installation problem. *SIAM Journal on Computing*, 38(6):2246–2442, 2009.

[4] S. Guha, A. Meyerson, and K. Munagala. Facility location with demand dependent costs and generalized clustering. *Manuscript*, 2000.

[5] S. Guha, A. Meyerson, and K. Munagala. Hierarchical placement and network design problems. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2000.

[6] M. Hajiaghayi, M. Mahdian, and V. Mirrokni. The facility location problem with general cost functions. *Networks*, 42:42–47, 2003.

[7] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1952.

[8] D. R. Karger and M. Minkoff. Building Steiner trees with incomplete global knowledge. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 613–623, 2000.

[9] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37(1):146–188, 2000.

[10] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Colloquium on Automata Languages and Programming*, pages 77–88, 2011.

[11] A. Lim, F. Wang, and Z. Xu. A transportation problem with minimum quantity commitment. *Transportation Science*, 40(1):117–129, 2006.

[12] M. Mahdian and M. Pál. Universal facility location. In *Proceedings of 11th ESA*, pages 409–421, 2003.

[13] P. Mirchandani and R. Francis, editors. *Discrete Location Theory*. John Wiley and Sons, Inc., New York, 1990.

[14] D. B. Shmoys. The design and analysis of approximation algorithms: facility location as a case study. In S. Hosten, J. Lee, and R. Thomas, editors. *Trends in Optimization, AMS Proceedings of Symposia in Applied Mathematics 61*, pages 85–97, 2004.

[15] D. B. Shmoys, É. Tardos, and K. I. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[16] M. Skutella. List scheduling in order of α-points on a single machine. In E. Bampis, K. Jansen, and C. Kenyon, editors. *Efficient Approximation and Online Algorithms: Recent Progress on Classical Combinatorial Optimization Problems and New Applications*, pages 250–291, Springer-Verlag, Berlin, 2006.

[17] M. Sviridenko. An improved approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of 9th IPCO*, pages 240–257, 2002.

[18] Z. Svitkina. Lower-bounded facility location. *Transactions on Algorithms*, 6(4), 2010.

[19] J. Zhang, B. Chen, and Y. Ye. A multi-exchange local search algorithm for the capacitated facility location problem. *Mathematics of Operations Research*, 30:389–403, 2005.

A Integrality-gap example for the natural LP-relaxation for CDUFL

Let $\bigl(\widehat{\mathcal{F}} = \widehat{\mathcal{F}}^u \cup \widehat{\mathcal{F}}^c, \widehat{\mathcal{D}}, \{\widehat{f}_i\}, \{u_i\}, \{\widehat{c}_{ij}\}\bigr)$ be a CDUFL instance with facility-set $\widehat{\mathcal{F}}$ (where $u_i = \infty$ for all $i \in \widehat{\mathcal{F}}^u$, and $\widehat{f}_i = 0$ for all $i \in \widehat{\mathcal{F}}^c$), and client-set $\widehat{\mathcal{D}}$. We consider the following LP-relaxation. We use i to index facilities, and j to index clients. Note that we may assume that all facilities in $\widehat{\mathcal{F}}^c$ are open.

$$\begin{aligned}
\min \quad & \sum_{i\in\widehat{\mathcal{F}}^u} \widehat{f}_i y_i + \sum_{j,i} \widehat{c}_{ij} x_{ij} && && \text{(LP)}\\
\text{s.t.} \quad & \sum_i x_{ij} \geq 1 && \text{for all } j \\
& x_{ij} \leq y_i && \text{for all } i \in \widehat{\mathcal{F}}^u, j \\
& \sum_j x_{ij} \leq u_i && \text{for all } i \in \widehat{\mathcal{F}}^c \\
& x_{ij}, y_i \geq 0 && \text{for all } i, j.
\end{aligned}$$

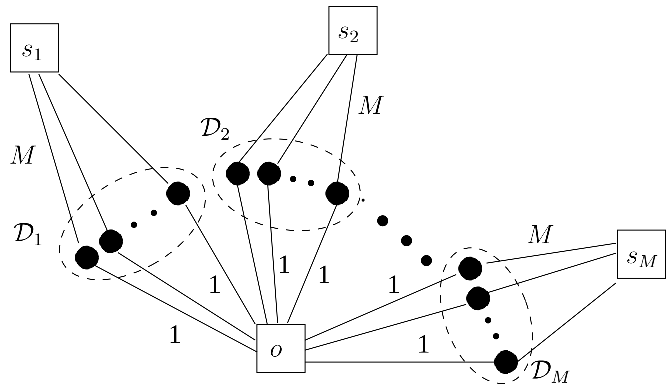
Here y_i denotes if facility i is open, and x_{ij} denotes if client j is assigned to facility i. (We assume that each client has unit demand.)

Now consider the following simple CDUFL instance. We have two facilities i and i', and $u+1$ clients, all present at the same location. Facility i is uncapacitated and has opening cost f, and facility i' has capacity u (and zero opening cost). Any solution to CDUFL must open facility i and therefore incur cost at least f. However, there is a feasible solution to (LP) of cost $\frac{f}{u+1}$: we set $y_i = \frac{1}{u+1}$, and $x_{ij} = \frac{1}{u+1}$, $x_{i'j} = \frac{u}{u+1}$. Thus, the integrality gap of (LP) is at least $u+1$.

B The locality gap of a local-search algorithm for LBFL

We show that the local-search algorithm based on add, delete, and swap moves—that is, adding/dropping one facility (with add permitted only if it preserves feasibility), or deleting one facility and adding another—has a bad *locality gap*, which is the maximum ratio between the cost of a locally-optimal solution and

that of an (globally) optimal solution. Consider the LBFL instance shown below with facility-set $\mathcal{F} = \{o, s_1, s_2, \ldots, s_M\}$, and client-set $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_M$, where the $\mathcal{D}_i$s are disjoint sets of size M. The facility-opening costs are as follows: $f_o = M^2 + \epsilon$ and $f_{s_i} = M$ for each $i \in \{1, 2, \ldots, M\}$. For each $i = 1, \ldots, m$ and each client $j \in \mathcal{D}_i$, we have $c_{oj} = 1,\ c_{s_i j} = M$. All other distances are defined by taking the metric completion with respect to these c_{ij}s. One can verify that the solution S which opens the facilities $\{s_1, s_2, \ldots, s_M\}$ is a local optimum. The cost of this solution is $M^2 + M^3$. However, the optimal solution opens facility $\{o\}$, and incurs a total cost of $2M^2 + \epsilon$. Thus, the locality gap is at least $M/2$.



We can modify this example to show that the locality gap remains bad, even if aim for a bicriteria solution and consider an add move to be permissible if every open facility can be assigned at least αM clients. The only change is that each set $\mathcal{D}_i$ now has αM clients: S is still a local optimum, and the locality gap is therefore at least $\alpha M/2$.

Bad example with zero facility-opening costs. Even in the setting where all facilities have zero opening cost (as in the $\mathcal{I}_2$ instance), we can construct bad examples for local-search based on add, delete, and swap moves. For simplicity, first suppose that $M = 2$. Consider a cycle with $4k$ nodes, which are labeled $o_0, j_0, s_0, j_1, o_1, j_2, s_1, j_3, \ldots, o_r, j_{2r}, s_r, j_{2r+1}, \ldots, o_{k-1}, j_{2k-2}, s_{k-1}, j_{2k-1}, o_0$. We have $2k$ facilities $\mathcal{F} = \{o_0, \ldots, o_{k-1}, s_0, \ldots, s_{k-1}\}$, and $2k$ clients $\mathcal{D} = \{j_0, j_1, \ldots, j_{2k-1}\}$ (see Fig. 3). We define the following distances.

- $c_{o_i j_{2i \bmod 2k}} = c_{o_i j_{(2i-1) \bmod 2k}} = 1$ for all $i = 0, \ldots k-1$.
- $c_{s_i j_{2i}} = c_{s_i j_{(2i+1)}} = k - \epsilon$ for all $i = 0, \ldots, k-1$.

All other distances are defined by taking the metric completion with respect to these c_{ij}s.

The solution S which opens facilities $\{s_0, s_1, \ldots, s_{k-1}\}$ is a local optimum: no add move is feasible, and it is easy to see that no delete move improves the cost. Consider a swap move, which we may assume is of the form $\mathrm{swap}(s_r, o_0)$ by symmetry. The new client-assignment will not necessarily assign the clients j_{2r} and j_{2r+1} (which were previously assigned to s_r) to o_0. However, the intuition is that the long cycle will lead to a large increase in assignment cost. The optimal way of reassigning clients is to assign j_{2k-1}, j_0 to o_o, assign j_{2i+1}, j_{2i+2} to s_i for $i \in \{0, \ldots, r-1\}$ (which is empty if $r = 0$), and assign j_{2i}, j_{2i-1} to s_i for $i \in \{r+1, \ldots, k-1\}$ (which is empty if $r = k-1$). The cost increase due to this reassignment is $2(1 - k + \epsilon) + (k-1) \cdot 2 > 0$. Thus, S is a local optimum.

The cost of S is $2k(k - \epsilon)$. However, the optimal solution opens facilities $\{o_0, \ldots, o_{k-1}\}$, and has a total cost of $2k$. So this instance shows a locality gap of k, and since k can be made arbitrarily large, this shows an unbounded locality gap.

The above example can be extended to all values of M. For each M, let G^M be an M-regular bipartite graph with vertex set $V = \{o_1, o_2, ..., o_\ell\} \cup \{s_1, s_2, ..., s_\ell\}$ with a large girth T. We use G^M to construct

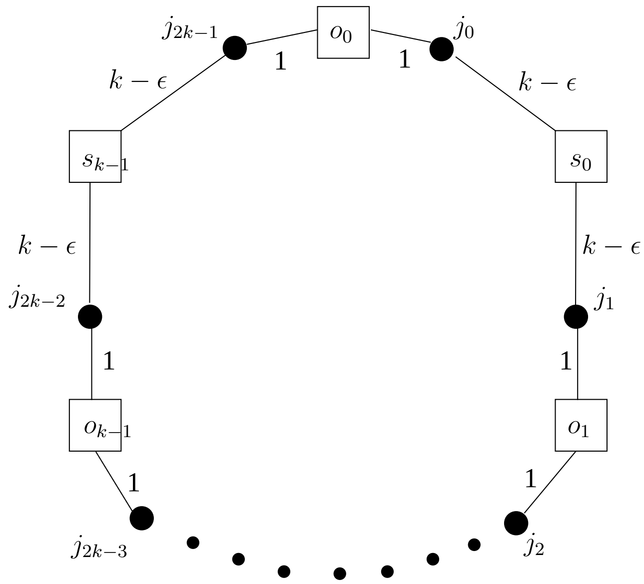


Figure 3: Bad locality-gap example with 0 facility costs

the following LBFL instance. The set of facilities is $\{o_1, \ldots, o_\ell, s_1, \ldots, s_\ell\}$. For each edge (s_n, o_m) in G^M, we create a client j_{nm} with $c_{s_n j_{nm}} = T - \epsilon$ and $c_{o_m j_{nm}} = 1$. As before, one can argue that the solution S that opens facilities $\{s_1, s_2, \ldots, s_\ell\}$ is a local optimum. The cost of this solution is $\ell M(T - \epsilon)$, whereas the solution that opens facilities $\{o_1, \ldots, o_\ell\}$ has total cost of ℓM. So the locality gap is T, which can be made arbitrarily large.